Convergence and rate of convergence of simple ant models

Amine Boumaza, Bruno Scherrer

Convergence and rate of convergence of simple ant models

Amine Boumaza and Bruno Scherrer

21/11/2007

Abstract

We present an original ant model to solve the foraging problem. We describe simulations and provide a convergence analysis. We prove the convergence of the model in the discrete and in the continuous cases. We show that the ant population computes the solution of an optimal control problem and converges in a well defined sense. We discuss the rate of convergence with respect to the number of ants for the discrete case: we give experimental and theoretical arguments that suggest that this convergence rate is superlinear with respect to the number of agents. Furthermore, we explain how this model can be extended in order to solve optimal control problems and more generally any problem that involves the computation of the fixed point of a contraction mapping. This allows us to design a large class of formally well understood ant-like algorithms for problem solving.

1 Introduction

Swarm intelligence [2, 5] was introduced in the 1990s as a novel nature-inspired approach for problem solving. The inspiring source is the behavior of real insects: a population of simple interacting agents, communicating indirectly through an environment, constitutes a *massively distributed* algorithm for solving a given task (e.g. foraging, flocking, labor division, prey capture, etc.). Referring to neural networks, another nature-inspired approach for massively distributed computing, Kohonen wrote in 1988 [13]:

> *One of the fundamental tasks of this new "neural networks" science is to demonstrate by mathematical analysis, computer simulations, and even working artificial sensory and control systems that it is possible to implement massively parallel information-processing functions using components, the principles of which*

are not mysterious but already familiar from computer technology, communication science, and control engineering. There is nothing in the "neural network" area which were not known, in principle at least, from constructs already in use or earlier suggested.

The motivation of this paper is to argue for a similar statement about swarm intelligence. Classical engineering problem solving and swarm intelligence can be viewed as alternative approaches of the same problem. Then what makes these approaches seem so different ? As Sutton pointed it [22] (when discussing the relations between "modern" Machine Learning and "classical" Intelligent Control), we could

characterize the split as having to do with the familiar dilemma of choosing between obtaining clear, rigorous results on the one hand, and exploring the most interesting, powerful systems on the other.

Roughly speaking, research on swarm intelligence is often focused on impressive proof-of-concept applications through extensive experimental simulations while classical engineering problem solving relies on theoretical convergence proofs for "toy" problems. The intent here is not to judge these approaches, which are clearly complementary, but rather to express our belief that filling the gap between both is a great research challenge.

There have not been many attempts in the literature that try to actually fill this gap, in particular in the area of ant-like algorithms. One interesting exception is the probably best known instance of swarm intelligence: the Ant-Colony Optimization (ACO) meta-heuristic for combinatorial problems, which has been extensively studied. The theoretical work on ACO have recently been reviewed in [10]. In this review, the authors present theorems for "convergence in value" (ACO is guaranteed to find an optimal solution in finite time with a probability that can be made arbitrarily close to 1) and "convergence in solution" (provided a properly designed decreasing exploration parameter, the ACO asymptotically converges to an optimal solution almost surely). They also relate ACO to "more standard" optimization techniques such as stochastic gradient descent and cross entropy. The interested reader should go through [10] for further details.

In this paper, we come back to the original, simpler, inspiration of ant algorithms, where a population of simple agents (that may be viewed as mimicking the behavior of real ants) efficiently solve a foraging problem.

Many models and algorithms have been proposed to solve this problem, see for example [8, 9, 16, 17, 19, 20]. These studies focus mainly on modeling the behavior of real ant colonies proposing algorithms and methods to solve interesting problems. However, from an engineering point of view, they do not go beyond simulations and do not provide formal analysis of the algorithms.

In this article, we introduce a massively distributed ant model that is guaranteed to asymptotically solve the foraging problem. We provide a convergence proof of the model, and a rate of convergence analysis with respect to the number of agents. Our analysis relies on optimal control theory, parallel distributed computing and graph theory. To our knowledge, this is the first attempt to present an ant model for the foraging problem and provide a formal analysis of the algorithm in terms of convergence *and* rate of convergence.

The remaining of the paper is organized as follows. Section 2 provides a description of our ant model and discusses some simulations that were made to measure the convergence and the rate of convergence. In Section 3 we give a formal proof of convergence and the theoretical analysis is used to discuss the influence of the model's parameters. Section 4 analytically discusses the rate of convergence: it gives formal arguments that explain why one observes a superlinear rate of convergence. In section 5, we describe the continuous version of the model we propose. We prove its convergence and illustrate through experiments the super-linearity of the rate of convergence with respect to the number of ants. Finally section 6 provides a discussion about the scope of our model and its relations to other models proposed in the literature.

2 A simple ant model

In this section, we describe a simple ant model, which is aimed at solving a foraging task on a discrete environment. We provide simulations that illustrate the behavior of the model and experimental measurements of its performance.

2.1 Description of the ant model

We consider a set of artificial ants moving on a two-dimensional grid environment[1] on which they update artificial *pheromone traces*. The environment is composed of four different types of cells: one *nest* cell, one *food* cell, several *bad* cells and all the remaining cells are considered *free*. Each cell s of the grid stores two pheromone traces as two real numbers: $\Phi_f(s)$ the food pheromone and $\Phi_n(s)$ the nest pheromone.

Ants can carry one unit of food at a time, and can be in two possible states: *carry food* or *carry nothing*. Food is picked up at food cells and dropped at the nest. When a unit of food is brought to the nest a food counter, which will serve as a global performance measure, is incremented. Initially:

- the food counter is set to 0,
- the positions of the ants are initialized arbitrarily (e.g. all ants are initialized at the nest or uniformly at random on the grid, etc.) and all ants are set to be in the *carry nothing* state,
- the pheromone values are initialized arbitrarily (e.g. 0, random, etc.)

At every time step, each ant performs two actions:

1. It updates both food and nest pheromone $\Phi_f(s)$ and $\Phi_n(s)$ of its current cell s using the pheromone values of its four neighbors (we note the set of neighbors of cell s as $\mathcal{N}(s)$), therefore using only local information. In fact, the update requires only the knowledge of the maximum and average of both pheromone values over the neighbors. Define:

$$\max_i(\mathcal{N}(s)) \equiv \max_{s' \in \mathcal{N}(s)} \Phi_i(s'),$$

and

$$\text{avg}_i(\mathcal{N}(s)) \equiv \frac{1}{4} \sum_{s' \in \mathcal{N}(s)} \Phi_i(s')$$

where $i \in \{f, n\}$ represents one of the *nest* or *food* pheromone. The pheromone update rules are then as follows:

$$\Phi_f(s) \leftarrow \begin{cases} 1 \text{ if } s \text{ is a } \textit{food} \text{ cell} \\ -1 \text{ if } s \text{ is a } \textit{bad} \text{ cell} \\ \beta \left[\alpha \max_f (\mathcal{N}(s)) \right. \\ \left. + (1-\alpha) \text{ avg}_f (\mathcal{N}(s)) \right] \text{otherwise} \end{cases} \tag{1}$$

[1] As the reader will understand in the following, more complicated graph structures could be used, we address here the 2D grid case for the sake of simplicity.

$$\Phi_n(s) \leftarrow \begin{cases} 1 \text{ if } s \text{ is a } \textit{nest} \text{ cell} \\ -1 \text{ if } s \text{ is a } \textit{bad} \text{ cell} \\ \beta\,[\,\alpha \max_n(\mathcal{N}(s)) \\ \quad +(1-\alpha)\ \text{avg}_n(\mathcal{N}(s))] \text{ otherwise} \end{cases} \tag{2}$$

where $0 \leq \alpha \leq 1$ and $0 \leq \beta \leq 1$ with the condition that $\beta < 1$ if $\alpha = 1$.

2. It moves to one of its four neighboring cells: with probability ϵ ($0 \leq \epsilon \leq 1$) (that we shall call the *exploration* rate) it moves uniformly at random to one of its neighbors, and with probability $1 - \epsilon$, it moves to the neighbor with the highest pheromone value. The internal state of the ants conditions which pheromone values are considered: Φ_f if it is in state *carry nothing* and Φ_n if it is in state *carry food*.

The β parameter can be considered as some sort of "evaporation parameter" and is typically to be set close to 1. The α parameter, which we shall call the noise parameter for reasons that will be explained later, should typically be set either close to 0 or close to 1. Two simple instances of our model correspond to the parameter choices ($\alpha = 0$, $\beta = 1$) and ($\alpha = 1$,$0 < \beta < 1$). In the former choice, the general pheromone equation (the "otherwise" case above) reduces to:

$$\Phi_i(s) \leftarrow \text{avg}_i(\mathcal{N}(s)), \tag{3}$$

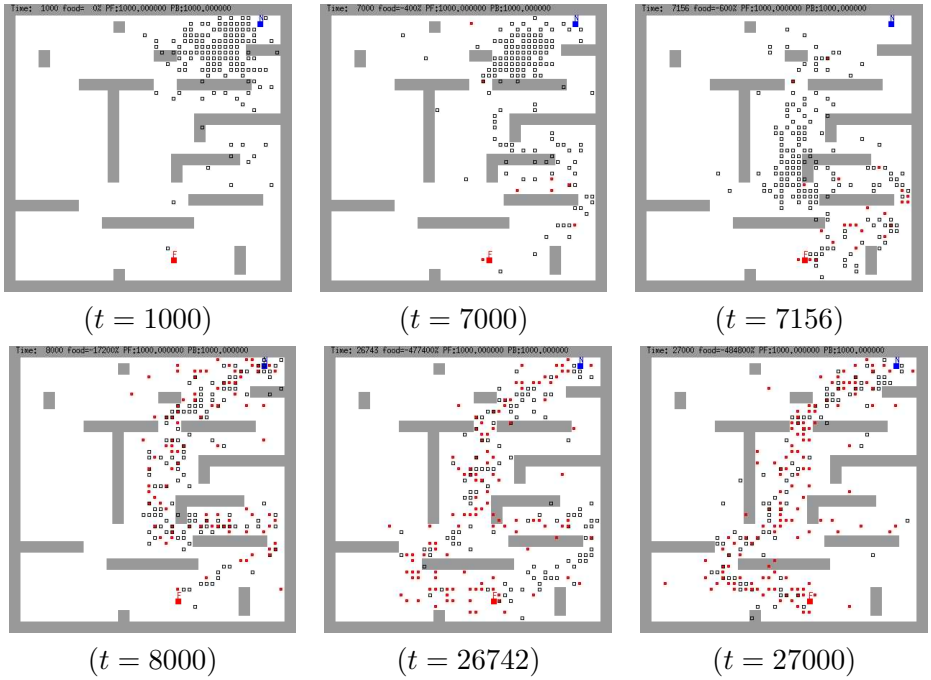
which is a simple linear update, and in the latter choice the equation reduces to:

$$\Phi_i(s) \leftarrow \beta \max_i(\mathcal{N}(s)). \tag{4}$$

As it will appear clearly in the analysis, the question whether the ant activities (pheromone values updates and moves) are done synchronously or asynchronously does not matter.

To summarize and to identify precisely different instances of this model (and for the reader to be able to reproduce the experiments we will soon describe), the following information needs to be specified:

- the environment: the set of cells, and their type (nest, food, free, bad),
- the way we initialize the position of the ants and the pheromone values Φ_f and Φ_n over the environment,
- the noise parameter α, the evaporation parameter β and the exploration parameter ϵ.



($t = 1000$) ($t = 7000$) ($t = 7156$)

($t = 8000$) ($t = 26742$) ($t = 27000$)

Figure 1: Snapshots of the ant model at different times. Thick gray lines represent undesirable (bad) states. The nest is located at the top right and the food source at the bottom in the middle (dark middle sized squares). Small squares represent the ants: filled squares represent ants carrying food, and empty squares are ants that are not carrying food.

The model we have just described is made of very simple reactive agents that communicate indirectly through the environment. In a way, this model is simpler than classical ant models: in our case, pheromones need not to be evaporated at *every* cell of the environment (this is a heavy computation when running a typical ant-like algorithm); in our model, some sort of evaporation is performed through the β parameter of the local pheromone update. However, unlike classical ant models which use one pheromone trail, ours uses *two*: following Φ_f leads to the food source and following Φ_n leads back to the nest. This is to compensate the fact that our ants, which are completely *reactive* agents, do not memorize the path to the nest. Previous work using a single pheromone trail often relies on ad-hoc mechanisms to help the ants return to the nest. Other ant models where two pheromone trails are used for foraging can be found in [12, 17, 24].

2.2 Simulations

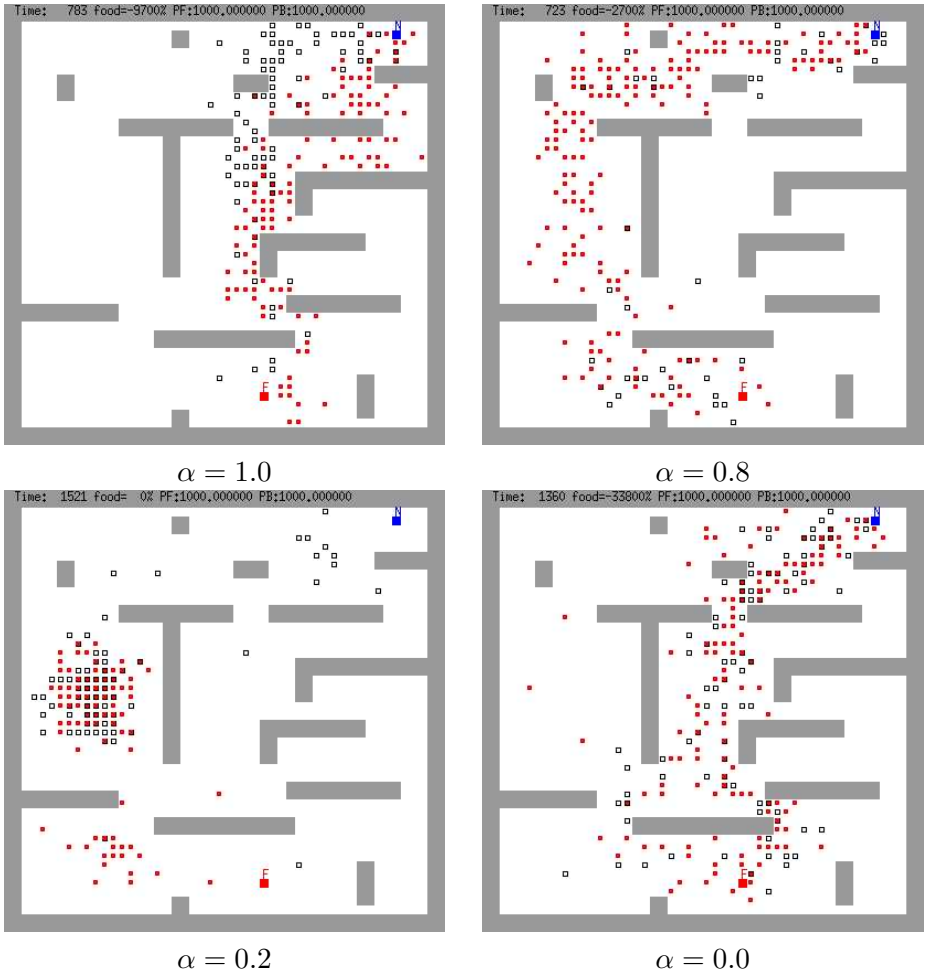
In this section we illustrate the behavior of our ant model with simulations. As we will see, we observe a form of convergence. We begin by describing what form of convergence we obtain and explain the experimental setting that allows us to measure the rate of convergence. Finally, we will briefly comment the results, as a formal in-depth analysis follows in sections 3 and 4.

Let us consider a typical run of the algorithm on the environment shown on Figure 1. In this simulation, the size of the population $m = 256$ and the ants were initialized at the nest. The other parameters of the algorithm were: $\epsilon = 0.75$, $\alpha = 0.99$, and $\beta = 0.9999$.

Figure 1 shows the evolution of the agents which, over time, move in the environment searching for food. The food source is discovered by chance by some isolated agents (iteration 7000). Quickly (between iteration 7000 and 8000), a large portion of the population starts following the discovered path. Later in the simulation, some agents find, while exploring, an alternative path (around iteration 26000) ; these agents are rapidly joined by the rest of the population (iteration 27000). At this stage, the dynamics stabilize and there will not be significant changes: the ants will move back and forth between the nest to the food source.

All other things being fixed, if we change the value of the noise parameter α, we can observe different forms of paths, and sometimes no path. Figure 2 illustrates the asymptotic distribution of ants for different values of α: we observe paths except for the case where $\alpha = 0.2$. Suppose a path emerges between the nest and the food source. We do observe such a path visually, however it would be interesting to measure something objective that reveals the emergence of this path. This can be done through the "food counter" we introduced previously. We may plot the quantity of food brought back to the nest with respect to time (see Figure 3). After some time, we observe that the quantity of food brought back to the nest increases linearly. This linearity means that the foraging behavior of the ants has converged. In section 3, we will provide theoretical arguments that characterize precisely this convergence, and we will in particular explain why we do not observe paths for some values of α.

The food counter curve shows experimentally for one simulation that there is convergence ; this curve can be exploited further to measure a convergence rate. For that we proceed as follows: we fix a sufficiently large maximum evolution time (t_{max}) and we fit a line on the upper portion of the food counter curve (the linear part of the curve). The size of the portion to fit is a parameter (p), which we typically chose between 0.25 and 0.5, taking



$\alpha = 1.0$ $\alpha = 0.8$

$\alpha = 0.2$ $\alpha = 0.0$

Figure 2: Limit distribution of the population for various values of α: different values of the noise parameter will result in different paths between the nest and the food source and sometimes none.

into account from the upper quarter to the upper half of the curve for the linear regression.

We define the *time of convergence* as the intersection of the fitted line and the time axis[2] (see Figure 3). The rate of convergence is thus defined as the inverse of the time of convergence.

[2]This is analogous to the computation of the time-constant of an electrical capacitor.

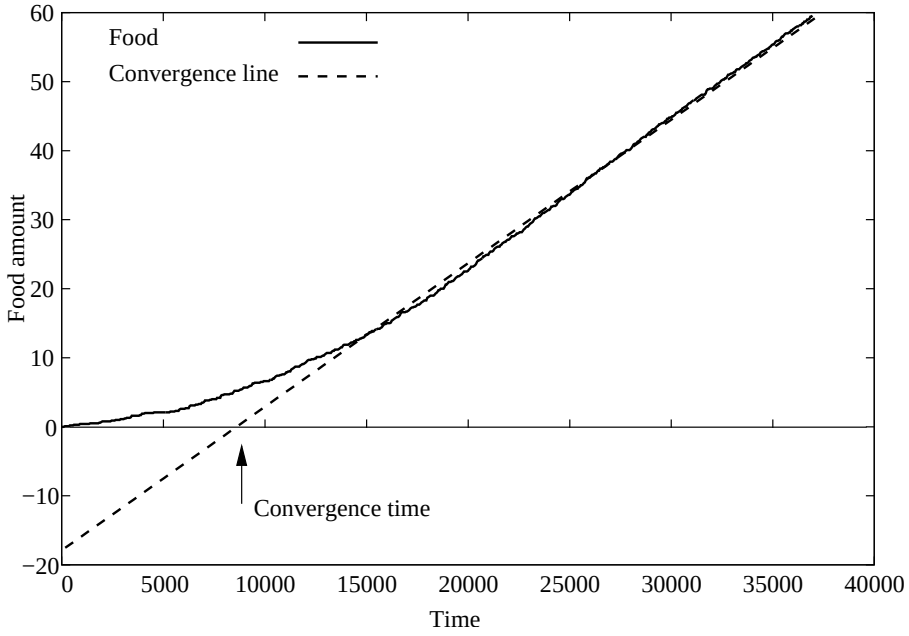


Figure 3: Measuring the rate of convergence: food units correspond to the accumulated amount of food brought back to the nest normalized by the number of ants. For one simulation we measure the rate of convergence as the inverse of the time when the convergence line intersects the x-axis (the convergence line is the linear regression over the linear part of the curve).

In order to determine the influence of the population size on the rate of convergence, we measure the rate for different values of the population size m. Thus one simulation consists in running the algorithm with different values of m on the same environment fixing all other parameters (α, β, ϵ, t_{max} and p). At the end of a simulation we obtain values of the rate of convergence with respect to m. Given the stochasticity of the simulation (the ants have ϵ probability of moving randomly), we reproduce the simulations a certain number of time in order to measure statistics on the rate of convergence.

Figure 4 presents a typical curve showing the rate of convergence with respect to the size of the environment. For this experiment, we used the environment of Figure 2 and the following parameters: $\alpha = 0.9$, $\beta = 0.999999$, $\epsilon = 0.7$, $t_{max} = 100000$ and $p = 0.5$. The agents were initialized uniformly at random over the environment. The statistics (median values and

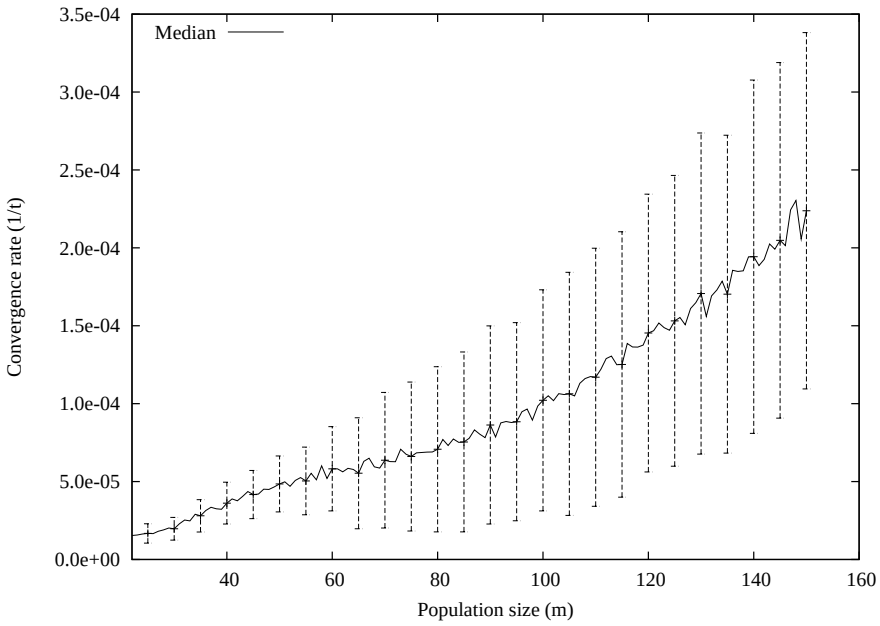


Figure 4: Convergence rate as a function of the population size: the curve represents the median value over 7000 simulations on the environment of Figure 2 with the following parameters: $\alpha = 0.9$, $\beta = 0.999999$, $\epsilon = 0.7$, $t_{max} = 100000$ and $p = 0.5$.

the absolute median deviation) were computed from 7000 simulations. Figure 5 shows similar experiments on different environments (with 5000 simulations).

These curves show experimentally that when the number of agents grows linearly, the convergence rate grows super-linearly. In other words, when the number of agents is doubled the rate of convergence grows with a factor greater than two. This observation will be analyzed in section 4: we will give arguments that support such a superlinear rate.

3 Convergence Analysis

In this part, we are going to give an analysis of our ant model. We will prove that it converges in some sense. To understand what our ant model does and why we see a path emerge, we will first make a detour into the theory of discrete-time stochastic optimal control, and particularly into the

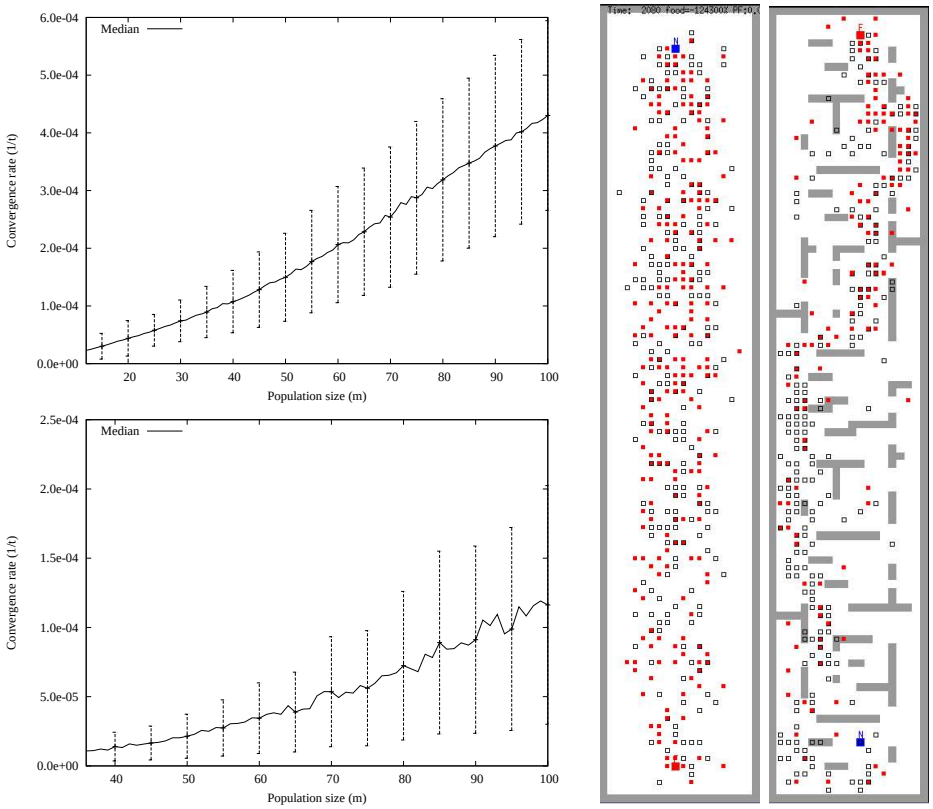


Figure 5: Convergence rate as a function of the population size: the curves represent the median value over 5000 simulations. The above curve corresponds to simulations in the environment on the left and the one below to the environment on the right. In both simulations, the parameters used were: $\alpha = 0.9$, $\epsilon = 0.8$, $\beta = 0.999999$, $t_{max} = 100000$ and $p = 0.5$.

Markov Decision Process (MDP) framework. A MDP is a controlled stochastic process satisfying the Markov property with rewards (numerical values) assigned to states. Formally, an MDP is a four-tuple $\langle S, A, T, R \rangle$ where S is the *state space*, A is the *action space*, T is the *transition function* and R is the *reward function*. T is the state-transition probability distribution conditioned by the action. For all states s and s' and actions a, $T(s, a, s')$ is the probability to go from state s to state s' after executing action a at state s,

$$T(s, a, s') \stackrel{def}{=} \Pr(s_{t+1} = s' | s_t = s, a_t = a).$$

$R(s) \in \mathbb{R}$ is the instantaneous reward for being in state s. In the MDP framework, the *optimal control problem* consists in finding a *policy*, that is a mapping $\pi : S \to A$ from states to actions, that maximizes the expected long-term amount of rewards, also called value function of policy π:

$$V^{\pi}(s) = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t . R(s_t) | s_0 = s\right]. \tag{5}$$

We here consider an infinite time horizon; also, future rewards are discounted exponentially with a discount factor $\gamma \in (0, 1)$ (setting $\gamma < 1$ can be seen as a mathematical trick so that the above performance criterion remains bounded). Given an MDP model, it is shown (e.g. by Puterman [18]) that there exists a unique optimal value function V^* which is the fixed point of the following mapping on functions, also called Bellman operator, $\forall W \in \mathbb{R}^n$:

$$[B.W](s) = \max_a \left(R(s, a) + \gamma . \sum_{s'} T(s, a, s') . W(s') \right) \tag{6}$$

Once an optimal value function V^* is computed, an optimal policy π^* can immediately be derived as follows:

$$\pi^*(s) \in \arg\max_a \left(R(s, a) + \gamma . \sum_{s'} T(s, a, s') . V(s') \right). \tag{7}$$

The fundamental reason why the above Bellman operator B has a unique fixed point is related to the fact it is a *contraction mapping* with contraction factor at most γ: i.e. for all pairs of real functions U, U' on S,

$$\|BU - BU'\| \leq \gamma \|U - U'\|$$

where $\|.\|$ is the "max-norm" on S: $\|U\| = \max_s |U(s)|$. A standard approach for solving the optimal control problem is to use a sequential iterative procedure, known as "Value Iteration" [18], which consists in initializing a value function V^0 arbitrarily and iterating the following process:

For all states $s \in S$, do:
$V^{t+1}(s) \leftarrow [BV^t](s)$.

Because of the contraction mapping property, this sequence is guaranteed to asymptotically converge to the optimal value function V^*, from which we can deduce the optimal controller π^* (cf equation 7). Furthermore, such an iterative sequence, has a linear rate of convergence γ:

$$\|V^{t+1} - V^*\| \leq \gamma.\|V^t - V^*\| \leq \gamma^{t+1}.\|V^0 - V^*\| \tag{8}$$

Berstekas and Tsitsiklis have argued [4] that an asynchronous version of Value Iteration:

$$\begin{array}{c} \text{Pick (randomly) a state } s \in S, \text{ do:} \\ V^{t+1}(s) \leftarrow [BV^t](s) \end{array} \tag{9}$$

will also converge to the fixed point V^*, *as long as all states keep on being picked*. In the asynchronous case, one can rewrite a variant of equation 8 as:

$$\|V^{k_t+1} - V^*\| \leq \gamma \|V^{k_t} - V^*\| \leq \gamma^{t+1}\|V^0 - V^*\| \tag{10}$$

where $k_0, k_1 \ldots$ is an increasing series such that $k_0 = 0$ and all components of s (all states) are updated at least once between instant k_t and $k_{t+1} - 1$ for all t (see [3] p. 27). Each time all the states have been updated, we know that V approaches V^* at a linear rate γ. In fact, once again, the proof of such a result relies on the contraction mapping property. We will in the following use this property to prove the convergence of our ant algorithm. While the aim in [4] was to come up with efficient parallel implementations on real parallel computers, ours is slightly different: we are going to show that the optimal control computation fits in the (virtual) ant paradigm.

Coming back to the ant model we described earlier, we are going to make a clear link between what the ant population computes and some optimal control problems. Indeed, we will show that the pheromone values Φ_f and Φ_n correspond each to the value function of a control problem. This is our first result:

Proposition 1 *Consider the ant model described in section 2.1. If the exploration rate ϵ is strictly positive, then the pheromone value Φ_f (resp. Φ_n) asymptotically converges to the optimal value function of the MDP $M_f = \langle S, A, T_f, R_f \rangle$ (resp. $M_n = \langle S, A, T_n, R_n \rangle$) with discount factor β where:*

- *S is the set of grid cell plus an extra "end state".*
- *A is the set of the four cardinal moves (North, East, South, West)*
- *The transition T_f (resp. T_n) is characterized as follows: 1) when, in a state s corresponding to a free cell or the nest cell (resp. to a free cell or the food cell), one chooses one of the four directions $a \in A$, the probability of actually making the move in the direction a*

is $\alpha + \frac{1-\alpha}{4} = \frac{3\alpha+1}{4}$ *while the probability of getting to each of the three other neighbors is* $\frac{1-\alpha}{4}$*. 2) From all the other states, that is from bad cells, the food and the "end state" (resp. from bad cells, the nest and the "end state"), there is, for every action, a probability 1 to reach the "end state", which is an absorbing state.*

- *The reward* R_f *(resp.* R_n*) is defined as follows: for all states* s *corresponding to a free cell or to a nest cell (resp. a free cell or a food cell), the reward is* 0*. For the state corresponding to the food cell (resp. the nest), the reward is* 1*. The reward is* -1 *for all states corresponding to bad cells and* 0 *for the "end state".*

Figure 6 provides a graphical view of what a part of these MDP can look like.

The proof of the above result simply consists in checking that the ant model is an asynchronous version of the value iteration algorithm for the corresponding problems: concretely, we need to check that, for all cells, the updates for Φ_f and Φ_n (equations 1 and 2) are identical to the one that would be done by the Bellman operator (equation 9). This is a simple question of rewriting. For instance, to update $\Phi_i(s)$ ($i \in \{f, n\}$) when s is a free cell, we have:

$$
\begin{aligned}
& \Phi_i(s) \leftarrow \beta\left(\alpha \mathrm{max}_i(\mathcal{N}(s)) + (1-\alpha)\mathrm{avg}_i(\mathcal{N}(s))\right) \\
\Leftrightarrow \quad & \Phi_i(s) \leftarrow \beta\left(\alpha \max_{s' \in \mathcal{N}(s)} \Phi_i(s') \right. \\
& \qquad \left. + \frac{1-\alpha}{4} \sum_{s' \in \mathcal{N}(s)} \Phi_i(s')\right) \qquad (11) \\
\Leftrightarrow \quad & \Phi_i(s) \leftarrow \beta \max_{s' \in \mathcal{N}(s)} \left(\alpha \Phi_i(s') + \frac{1-\alpha}{4} \sum_{s'' \in \mathcal{N}(s)} \Phi_i(s'')\right) \\
\Leftrightarrow \quad & \Phi_i(s) \leftarrow \beta \max_{s' \in \mathcal{N}(s)} \left(\left(\alpha + \frac{1-\alpha}{4}\right) \Phi_i(s') \right. \\
& \qquad \left. + \frac{1-\alpha}{4} \sum_{s'' \in \mathcal{N}(s) \setminus \{s'\}} \Phi_i(s'')\right) \\
\Leftrightarrow \quad & \Phi_i(s) \leftarrow \beta \max_{a \in A} \left(\sum_{s' \in S} T_i(s, a, s') \Phi_i(s)\right). \qquad (12)
\end{aligned}
$$

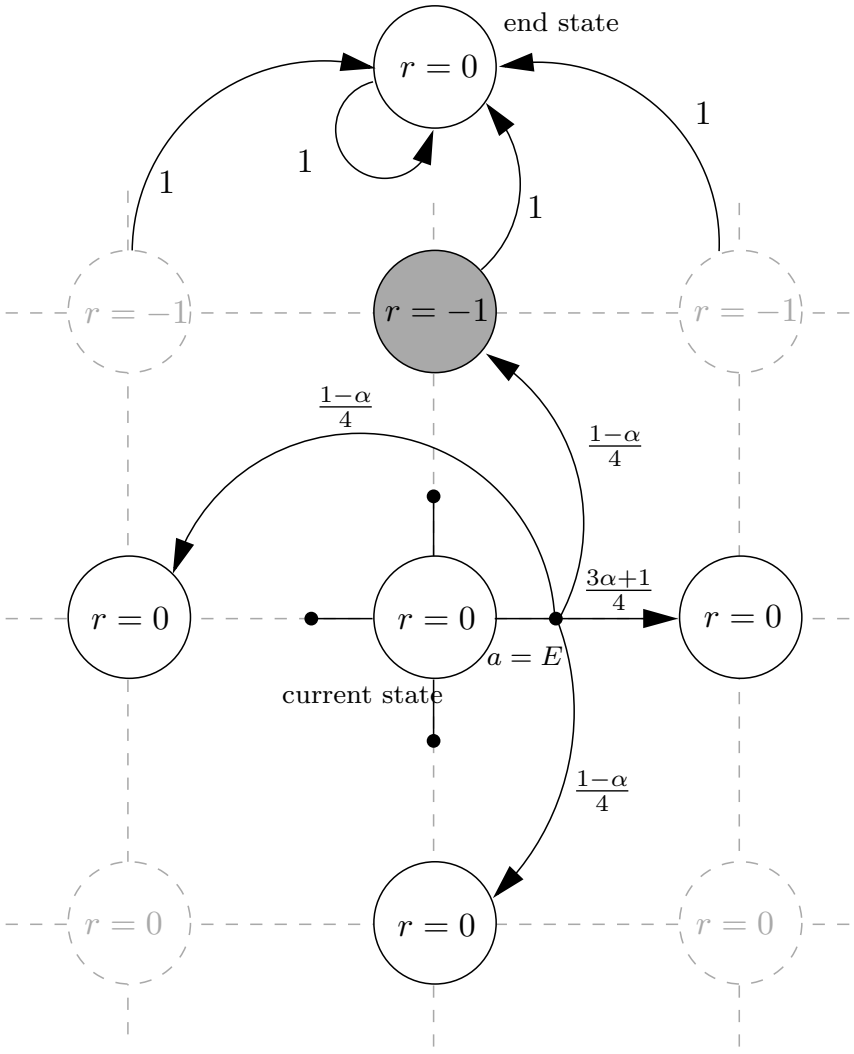


Figure 6: Representation of the MDP model. Each node of the graph is a state. The associated reward is written inside the node. In one state, there are four actions. The picture above shows the transition probabilities from the current state (the center cell) if action East is picked. It also shows the transitions to the "end state". See text for details.

We let the interested reader check that this equation also holds in the few other cases (when s is not a free cell). Finally, the condition $\epsilon > 0$ ensures that all states will keep on being visited and updated, which in turn ensures the convergence of this asynchronous version to the optimal value functions.

Now let us interpret what this means and especially why we observed the emergence of paths between the nest and the food source in the experiments. Solving M_f (resp. M_n) means finding a policy that will generate trajectories that avoids (on average) the bad cells for which the reward is -1 and try to reach the food cell (resp. the nest cell) for which the reward is 1; because of the discount factor $\beta < 1$, the optimization also tries to minimize the time to reach the food cell (resp. the nest cell). In other words, M_f (resp. M_n) are natural formulations of the control problem: "go to the food cell (resp. the nest cell) by avoiding the bad states" assuming that there is some noise in the transition models T_f and T_n. The level of noise is related to α: calling α a "noise parameter" was thus fully justified.

In each state, the optimal action is the one for which the maximum is reached in equation 12 above, and it is easy to see that this optimal action is the one that points to the cell for which the maximum is reached in equation 11: that is the cell with the highest pheromone value. The pheromone values asymptotically converge to the corresponding optimal value functions Φ_f and Φ_n. Therefore, the moves of the ants, which are (recall the ant move description in section 2.1) a mixture of random uniform moves (with a weight ϵ) and the action that climbs up the pheromone value (with a weight $1 - \epsilon$), converge to a mixture of random uniform moves and the optimal corresponding moves. The smaller ϵ, the clearer will be the path when the pheromones have converged (at the limit if $\epsilon = 0$ all the ants follow the optimal policy). However, the smaller ϵ, the longer it will take for the ants to repeatedly visit all the states and make the pheromone values converge. This trade-off is typical of optimal control theory and is known as the exploration-exploitation dilemma (see for instance Sutton & Barto [23]).

In the description of the algorithm in section 2.1, we wrote about the parameters α and β that we needed $0 \leq \alpha \leq 1$ and $0 \leq \beta \leq 1$ with the condition that $\beta < 1$ if $\alpha = 1$. We can now explain these conditions from the optimal control viewpoint: setting the discount factor of an MDP to a value strictly lower than 1 ensures that the infinite time horizon performance (equation 5) remains bounded and that the Bellman operator is a contraction mapping. As soon as there is some amount of noise (i.e. $\alpha < 1$) in the transition model of our MDP M_f and M_n, then for any action, there is a probability 1 of reaching one of the absorbing "end state" (with zero reward) and this suffices to ensure that the performance criterion remains

bounded and the Bellman operator is contracting. However, in the purely deterministic case ($\alpha = 1$), the discount factor β needs to be set to a value strictly lower than 1.

We can explain further the influence of the noise parameter α. When α is equal to 1 (and the update is equation 4), there is no noise in the transition model and the optimal policies matches the shortest path (with the respect to the Manhattan distance, see Figure 2, case $\alpha = 1$) between the nest and the food source. When we decrease α, the level of noise increases and the optimal policies get smoother (they try to stray away from the bad states). At some point, when α goes on decreasing, there is so much noise that the probability of first reaching a bad state when trying to reach the food source (or the nest) is too large for any policy. Consequently the optimal behavior consists in simply staying away from the bad cells and not trying to reach the food source (or the nest): in this case it is better to have a 0 reward than a -1 (which happens when one hits a bad state). This explains why there was no path in Figure 2 for $\alpha = 0.6$. Furthermore when α goes on decreasing and gets close to 0, something apparently strange happens: paths appear again. We can give two explanations of this: 1) when the noise gets so large that the influence of the actions nearly vanish, the optimal controller cannot even prevent from hitting a bad state and, as a "kamikaze" who would know he's going to die anyway, it again becomes interesting to try to reach the food source (or the nest). 2) At the limit when $\alpha = 0$ (and the update is equation 3), the pheromone potentials, which satisfy an equation of the type $\Phi_i(s) = \text{avg}_i(\mathcal{N}(s))$ is equal to a discrete harmonic function and it is known that climbing a harmonic function can be used for navigation (see Connolly [7] and Boumaza & Scherrer [6] for further details).

4 Rate of convergence analysis

We showed the convergence of the proposed ant model by arguing that it is an asynchronous computation of two contraction mapping fixed points: pheromone potentials that result from the ant local updates stabilize towards the optimal value functions of some control problems, which guide the ants between the nest and the food source. The aim of this section is to study the *rate of convergence* of this process with respect to the number of ants. We shall describe some objects and results related to Markov chains on graphs that highlight the experimental observation (made in section 2.2) that this rate of convergence is superlinear: doubling the number of ants may accelerate the process by more than a factor two.

To study the rate of convergence, we go back to equation 10 that describes, in general, the convergence rate of the asynchronous computation of a contraction mapping fixed point. We rewrite it here for the sake of clarity:

$$\|V^{k_t} - V^*\| \leq \gamma \|V^{k_{t-1}} - V^*\| \leq \gamma^t \|V^0 - V^*\| . \tag{13}$$

Recall that $k_0, k_1 \ldots$ is an increasing series such that $k_0 = 0$ and all components of s (all states) are updated at least once between instants k_t and $k_{t+1} - 1$ for all t. The rate of convergence is thus clearly related to the random variable k_t: the slower k_t grows the faster the process converges. Since we are interested in the rate with respect to the number m of ants, we can make this dependence explicit by writing k_t^m. What we need to study is thus $k_{t+1}^m - k_t^m$.

Fortunately the quantity $k_{t+1}^m - k_t^m$ can be related to a known object of the probability literature. Consider the expectation of this quantity when $m = 1$: $E[k_{t+1}^1 - k_t^1]$ is the average time for one ant to visit all the cells of the environment. More formally, if we see the environment as a graph G (there is one node for each cell and a connection when two cells are neighbors) it is the average time for a Markov chain (that describes the positions of the ant) to visit all the nodes of the graph G, and such a quantity is usually called the *cover time of the Markov chain on the graph* G [1]. Similarly, for any m, $E[k_{t+1}^m - k_t^m]$ is the average time for several parallel Markov chains to visit all the nodes of the graph G and it is known as the *cover time of the parallel Markov chains on the graph* G.

Unfortunately, it is *in general* very hard to compute the cover time of a graph for a given Markov chain and a given initial distribution: they are computable in exponential time and it is not known whether it is possible to approximate them in deterministic polynomial time [11]. It is thus reasonable to think that it might be harder to compute the cover time of a graph for several Markov chains. For our specific ant model, it is probably even harder, since we would need to compute the cover times since the Markov chains and the distribution of ants vary over time in a non-trivial way: the transition probabilities depend upon the pheromone potentials which themselves are continuously updated by the Markov chains. There is therefore little hope that one could estimate *very precisely* the rate of convergence of our ant model. A general asymptotic study of the convergence (where one considers that 1) the pheromone values have converged and 2) the ants have reached their stationary distribution) may be pursued and this is a possible subject for future research.

Nonetheless, by studying the literature on the cover time, we were able to find the following interesting result by Aldous [1]:

Proposition 2 *On a regular n-vertex graph*[3]*, consider m independent balanced random walks*[4]*, each started at a uniform random vertex. Let* C^m *be the time until every vertex has been hit by some walk*[5]*. Then as the size of the graph* $n \to \infty$ *and while the number of walks satisfies* $m \geq 6 \log n$*, we have:*

$$E\left[C^m\right] \leq \frac{(25 + o(1))n^2 \log^2 n}{m^2}.$$

The interesting point is the $\frac{1}{m^2}$ dependence: this implies that (as $n \to \infty$ and $m \geq 6 \log n$) the cover time is bounded by a function which is inversely quadratic in the number of walks. On simple graphs like the $n - cycle$, Aldous explains that the above bound is sharp, so in such a case, the cover time is inversely quadratic in the number of walks [1].

Using the above discussion, we can "translate" the above proposition into something that is meaningful for our ant framework:

Proposition 3 *Consider the ant model described in section 2.1 on a toric grid environment with n cells, with an exploration rate* $\epsilon = 1$ *and initialize the ants uniformly on the environment. When the size of the environment* $n \to \infty$ *and while the number m of ants satisfies* $m \geq 6 \log n$*, the time for the pheromone values to reduce their distance to their limit by a factor* β *is bounded by* $\frac{(25+o(1))n^2 \log^2 n}{m^2}$ *that is a function that is inversely quadratic in the number m of ants.*

We consider a *toric* environment so that the corresponding graph is regular. We take $\epsilon = 1$ so that the ants follow a balanced random walk. Also notice then that the uniform initialization is the stationary distribution of the random walks, so that the distribution of ants is stationary over time. Our proposition is thus simply a corollary of proposition 2.

The bound is sharp when the environment is a n-cycle graph, that is a long (cycled) corridor: in such a case, the rate of convergence can be quadratic in the number of ants. We there have a superlinear rate of convergence. The experiments we made suggest that superlinearity also happens for other values of ϵ and general environments. Extending the above results to these more general setting constitutes future research.

[3] A regular graph is a graph where all nodes have the same degree, i.e: each node is connected to the same number of nodes.

[4] A balanced random walk on the graph is such that transitions are uniform distributions on neighbors.

[5] $E[C^m]$ is thus the cover time of the graph by these parallel random walks.

5 Extension to continuous space

The fact that the environment we have considered so far was *discrete* might seem like a strong limitation for the scope of our ant model. We aim in this section to show that, with slight modifications, most of the ideas described previously apply in a continuous (space & time) foraging task where ants can move in any direction $\theta \in (0, 2\pi)$.

Our approach will be rather straight-forward: we will describe a continuous variant of navigation through optimal control, argue that it can be approximated by an asynchronous distributed algorithm, and deduce the corresponding ant algorithm. Though in this article our presentation of the discrete model was made the other way round (we presented the ant model and *then* the link with optimal control), this section will show the methodology we actually followed for building it. For the sake of simplicity, we will here restrict our attention to half of the foraging task: finding paths to some food source. The other half can be implemented in the same way as we did for the discrete case (using an internal state for each ant and switching between two optimal control problems "going to food"/"going back to the nest").

A natural model for navigation in a continuous 2D setting goes as follows. One considers a system defined at time t by its position in the free environment (or equivalently its state) $x(t) \in \bar{\Omega}$ where $\bar{\Omega} \subset \mathbb{R}^2$ (the state space) is the closure of an open set Ω and $\partial\Omega$ is its boundary ($\bar{\Omega} = \Omega \cup \partial\Omega$). The boundaries of the environment are decomposed into two sets $\partial\Omega = \mathcal{B} \cup \mathcal{F}$: $\mathcal{B}$ is the set of bad states and $\mathcal{F}$ the set of food states. At each time step, this system is controlled by a direction variable $\theta(t) \in (0, 2\pi)$ ($(0, 2\pi)$ is the control space). The dynamics of such a system is governed by the following stochastic differential equation:

$$dx = \overrightarrow{u}(\theta(t))\, dt + \sigma\, dw$$

where w is a 2-dimensional Brownian motion, σ is a positive constant that corresponds to the level of noise in the environment, and $\overrightarrow{u}(\theta)$ is a unit vector in the direction $\theta \in (0, 2\pi)$ (thus a unit speed control). We consider the case of infinite time horizon. For any initial state x_0, any control law $\theta(.)$ and trajectory $x(.)$, we note τ the exit time of $x(.)$ from Ω, with the convention that $\tau = \infty$ when the trajectory stays infinitely within Ω. We use rewards in order to evaluate the performance of these trajectories: one defines an instantaneous reward function $r : \Omega \to \mathbb{R}$ and a terminal reward function $R : \partial\Omega \to \mathbb{R}$ when the system hits a boundary. For our simple navigation purpose, we take $r(x) = 0$ at free states ($\forall x \in \Omega$), $R(x) = -1$

on bad boundaries ($\forall x \in \mathcal{B}$) and $R(x) = 1$ on food boundaries ($\forall x \in \mathcal{F}$). Similarly to the discrete case, we introduce an optimal value function which is the maximum expected total amount of discounted reward on the controlled trajectories:

$$\begin{aligned} J^*(x) &= \max_{\theta(.)} E_{\theta(.)}\left[\int_0^\tau \gamma^t r\left(x\left(t\right)\right)\, dt + \gamma^\tau R\left(x\left(\tau\right)\right)\right] \\ &= \max_{\theta(.)} E_{\theta(.)}\left[\gamma^\tau R\left(x\left(\tau\right)\right)\right] \end{aligned}$$

where the integral term vanishes since the instantaneous reward $r(\cdot)$ is 0. As the function $t \mapsto \gamma^t$ is decreasing and because of the definition of the terminal reward function (-1 on bad boundaries and 1 on food boundaries), maximizing this quantity intuitively means both 1) maximizing the time of hitting an obstacle and 2) minimizing the time of reaching food.

The theory of optimal control shows that the above value function satisfies a partial differential equation known as the Hamilton-Jacobi-Bellman equation:

$$\begin{aligned} J^*(x)\,\ln(\gamma) &+ \max_{\theta\in(0,2\pi)}\left\{\nabla J^*(\,x\,)\cdot\overrightarrow{u}(\theta)\right\} \\ &+ \frac{\sigma^2}{2}\,\Delta J^*\left(x\right) = 0 \end{aligned} \tag{14}$$

for $x \in \Omega$ with boundary conditions $\forall x \in \partial\Omega, J^*(x) = C(x)$, where ∇J^* denotes the gradient of J^* and ΔJ^* the Laplacian of J^*. Analogously to the discrete case, the optimal control (the optimal direction θ) is the one for which the max in equation 14 is attained. In this case, it is easy to see that the optimal control $\theta^*(x)$ in state x is the direction of steepest ascent of $\nabla J^*(x)$: $\overrightarrow{u}(\theta^*(x)) = \frac{\nabla J^*(x)}{\|\nabla J^*(x)\|}$. Indeed, $\nabla J^*(x) \cdot \overrightarrow{u}(\theta)$ is maximal when θ is in the direction of the gradient $\nabla J^*(x)$.

In practice, one cannot compute exact analytical solutions to equation (14) and a usual approach consists in building a finite difference scheme. To do so, we follow the lines of Kushner & Dupuis [14]. Given a space discretization resolution $\delta > 0$, we build a grid Σ^δ and its border $\partial\Sigma^\delta$ on the domain of the problem. Given a grid resolution δ, the function J is approximated by a function J^δ defined on $\Sigma^\delta \cup \partial\Sigma^\delta$ and the optimal controller is the one that goes up the gradient of an interpolation (e.g. a triangulation) of J^δ on Ω.

According to the theory of optimal control [14], this discretization ends up by some good news: the (discrete-space) approximation J^δ of the continuous value function J^* happens to be the value function of some (discrete-time) optimal control problem. In other words, we fall back on an MDP. As

a consequence, it is straight-forward to build an ant-like model that computes J^δ as an asynchronous version of the Value Iteration algorithm. The pheromone updates just need to match the corresponding Bellman operator, which we describe now.

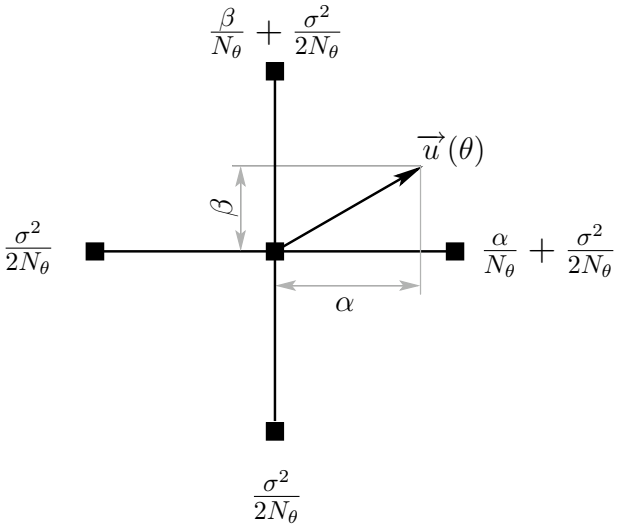


Figure 7: Transition probabilities $p_\theta\left[(x,y),(x',y')\right]$ in the discrete model: neighbors in direction θ have a component that is proportional to the coordinates of the unit vector $\overrightarrow{u}(\theta)$.

We define $\cos^+$, $\cos^-$, $\sin^+$ and $\sin^-$ as the positive and negative parts of cos and sin: $\cos^\pm(\theta) = \max(\pm\cos\theta, 0)$ and $\sin^\pm(\theta) = \max(\pm\sin\theta, 0)$. Furthermore, we define the following transitions probability from grid point (x,y) to its four neighbors when going in direction θ:

$$\begin{cases} p_\theta\left[(x,y),(x\pm\delta,y)\right] = \frac{1}{N_\theta}\left[\frac{\sigma^2}{2} + \delta\cos^\pm\theta\right] \\ p_\theta\left[(x,y),(x,y\pm\delta)\right] = \frac{1}{N_\theta}\left[\frac{\sigma^2}{2} + \delta\sin^\pm\theta\right] \end{cases} \tag{15}$$

where $N_\theta = \delta(\cos^+\theta + \cos^-\theta + \sin^+\theta + \sin^-\theta) + 4\sigma^2/2 = \delta(|\cos\theta| + |\sin\theta|) + 2\sigma^2$ can be regarded as a normalizing factor that ensures that the the transition probabilities $p_\theta((x,y),(x',y'))$ sum to 1. These transition probabilities on the discretized grid have a natural geometric interpretation: all of them have the same noise component ($\frac{\sigma^2}{2N_\theta}$), and two of them (the ones that are in the control direction θ) have non-zero weights (see Figure 7). We finally write $\tau(\theta) = \frac{\delta^2}{N_\theta}$, which is a quantity that can be interpreted as the time needed to go from one grid point to another when following direction

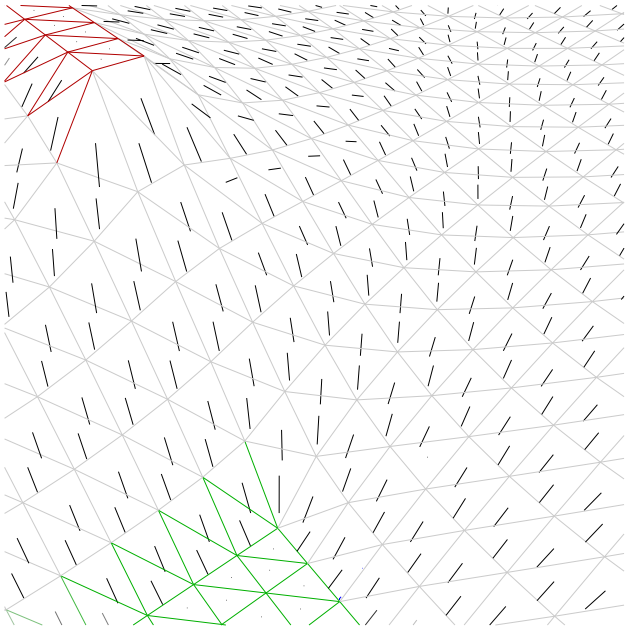


Figure 8: A discrete representation of J^δ, illustrating the gradients vector field.

θ. Then, an approximation of J is obtained by computing the unknown J^δ in the following system:

$$J^\delta(x,y) = \gamma^{\tau(\theta^{J^\delta}_{x,y})} \sum_{(x',y')\in\Sigma^\delta} p_{\theta^{J^\delta}_{x,y}}\left[(x,y),(x',y')\right]\, J^\delta(x',y'), \tag{16}$$

for all $(x,y) \in \Sigma^\delta$ and $J^\delta = C$ on $\partial\Sigma^\delta$, and where $\theta^{J^\delta}_{x,y}$ is the angle that corresponds to the steepest slope direction at (x,y) when considering a piecewise linear interpolation of J^δ (see Figure 8). The notation $\theta^{J^\delta}_{x,y}$ may seem somewhat heavy, but it is important to remember that the optimal angle depends on the coordinate (x,y) and the value function J^δ. Introducing the following operator B^δ on Σ^δ:

$$[B^\delta\, W](x,y) \;=\; \gamma^{\tau(\theta^{W}_{x,y})} \sum_{(x',y')\in\Sigma^\delta} p_{\theta^{W}_{x,y}}\left[(x,y),(x',y')\right]\, W(x',y')$$

for all $(x,y) \in \Sigma^\delta$ and $[B^\delta\, W] = C$ on $\partial\Sigma^\delta$, thus equation 16 becomes $J^\delta = B^\delta[J^\delta]$. Since $\gamma < 1$ and $\tau(\cdot) > \frac{\delta^2}{\delta+2\sigma^2} > 0$, the operator B^δ satisfies a contraction property (with contraction factor at least $\gamma^{\frac{\delta^2}{\delta+2\sigma^2}}$). Therefore, J^δ

is unique, and can be computed using value iteration, that is as the limit of the sequence $J^{\delta}_{t+1} \leftarrow B^{\delta}[J^{\delta}_{t}]$ when t tends to infinity. Once J^{δ} is computed, the (approximate) optimal control is the direction $\theta^{J^{\delta}}_{x,y}$ that descends the gradient of (the linear interpolation of) J^{δ}. It can be proved [14] that such a finite difference scheme is convergent: J^{δ} uniformly tends to J^{*} and the approximate optimal controller ∇J^{δ} tends to the optimal controller ∇J^{*} when the discretization resolution δ tends to 0.

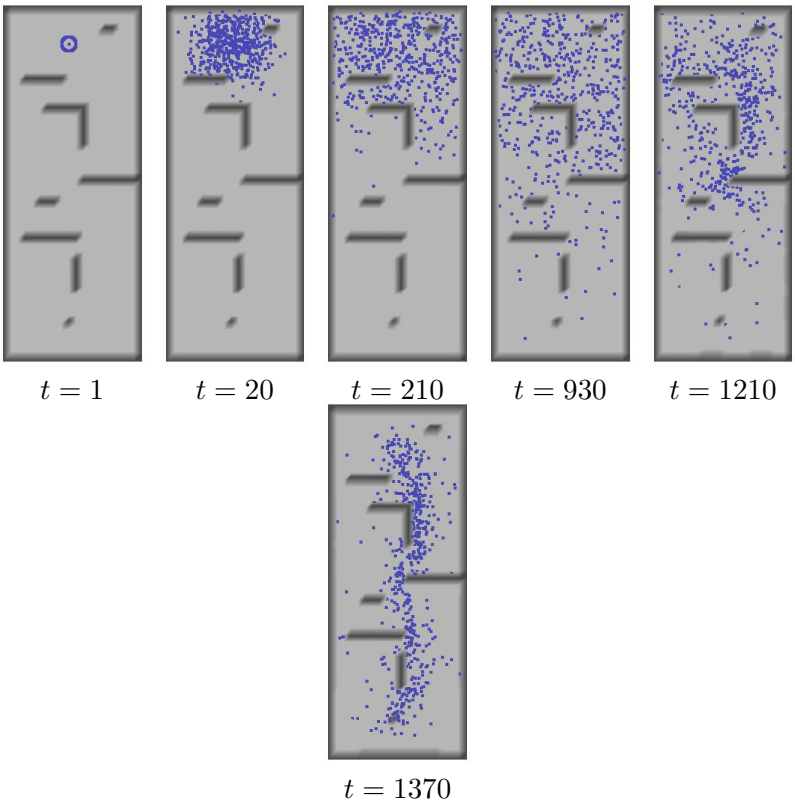


Figure 9: A typical experiment showing the convergence and the path emergence: four snapshots of an execution of the algorithm on a continuous environment underlying 20×50 grid point after discretization and 256 ants. See the text for details.

At this stage, a continuous instance of the ant algorithm can now be easily built. Ants are embedded in the original continuous environment (each ant has continuous coordinates) ; they move around and update the pheromone value which is coded by J^{δ}. At each time step, each ant does

two things:

1. It updates the pheromone value $J^\delta(x)$ of the closest grid point x as follows:
$$J^\delta(x) \leftarrow [B^\delta J^\delta](x)$$
therefore using only the pheromone values of x's neighbors.

2. It makes a move of a given length $l \in \mathbb{R}$: with probability ϵ (the *exploration* rate) it moves in a direction that is chosen uniformly at random in $(0, 2\pi)$. With probability $1 - \epsilon$, it moves in the direction of the steepest slope of the (local) linear interpolation of J^δ (following the optimal controller with respect to J^δ). If an ant reaches a food border, it is reinitialized at the nest.

A typical simulation of this algorithm is illustrated in Figure 9, in an environment with one nest one food source. Ants are all initialized at the nest. One observes the emergence of a path that is being reinforced as long as time goes.

5.1 Simulations

We follow the same experimental protocol as presented in section 2.2 and measure the rate of convergence with respect to the size of the population in the continuous setting. Since we have only addressed one MDP, the one whose solution will lead to the food source, we will increment the food counter each time an ant reaches the food source.

In this case, it is also possible to make a convergence analysis. If the exploration rate $\epsilon > 0$, one can prove that the ants will keep on passing by and updating every grid point. The population therefore constitutes an asynchronous version of Value Iteration. Thus similarly to the discrete case the algorithm converges.

Furthermore the experimental results using this continuous version exhibit a superlinearity of the convergence rate as shown in Figure 10. We can also characterize the rate of convergence using some generalized notion of cover time. In the case where ants move in the real continuous environment while updating the value of the closest grid points, the dynamical process of the grid points that are updated is not a simple Markov chain, but a hidden Markov chain: the specific grid point that is updated is a function of the real position of the ant, which is itself a Markov chain on the continuous environment. We should here consider the cover time of some hidden Markov

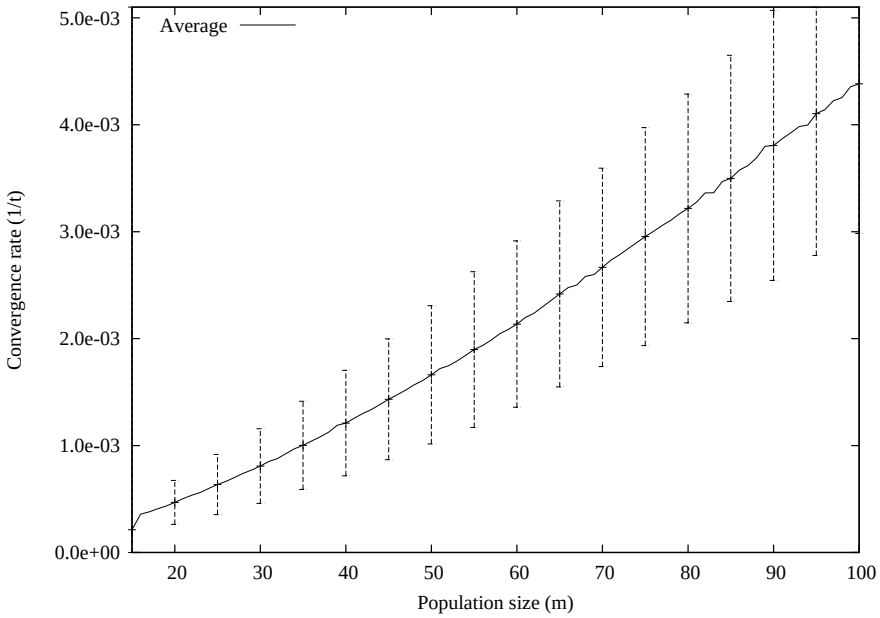


Figure 10: Convergence rate as a function of the population size: the curve represents the mean value over 5000 simulations on the environment of Figure 9 with the following parameters: $\sigma = 0.01$, $\beta = 0.99$, $l = 1$, $\epsilon = 0.7$, $t_{max} = 10000$ and $sp = 0.5$.

chain. Unfortunately, to the best of our knowledge, this quantity has not been studied in the probability literature. If we have been able to observe experimentally a superlinear rate of convergence for the continuous model, we have no theoretical result that would support it.

6 Discussion

We have presented a class of ant models that can be related to the framework of optimal control, and proved that they converge in some sense. We have also studied the rate of convergence with respect to the number of ants: we showed, through experimental and analytical arguments, that the rate of convergence is superlinear in the number of ants. At first sight, superlinearity may look like an impressive property. Nothing actually comes for free: the fundamental reason why we have such a superlinear rate is due to the fact that small populations of ants are *especially* inefficient at visiting

the whole state space, and therefore making the pheromone potentials converge. In fact, the convergence analysis of contraction mapping that we used clearly suggests (compare equations 8 and 10) that the fastest method for computing the optimal value function is the synchronous version: at each step, the value is approaching its limit with the linear rate γ. The relative slowness of the asynchronous version has to be understood as the price for a decentralized update of the pheromones.

There are many directions in which the current work could be extended. The ant model is very flexible and can incorporate many variations. For instance, we could consider that there are several food sources, each containing a finite quantity of food, which decreases over time as ants come and go. In this case the quantity could be incorporated in the corresponding optimal control model, through the reward function at food states. The reward would then evolve over time, and the pheromones, which are continuously being updated, would keep on following the corresponding "moving" optimal value function. This idea could be extended even further, including moving food sources as suggested by [17]. One could also use different parameters for the back and forth trips of the ants: this could model different strategies depending on the fact that the ant carries food or not. Finally, if the environment is static, one could make the exploration rate ϵ slowly tend to 0, so that the ants eventually converge to the deterministic optimal policies. Studying good ways of "freezing" the ant behavior through the exploration parameter ϵ constitutes future research.

It is in fact easy to see that one could construct similar ant algorithms for almost any optimal control problem. As long as the problem is formulated as an MDP, we know that it can be solved by an asynchronous version of Value Iteration, and building the corresponding ant system is immediate: we just need to have ants move around the state space and do the Bellman update everywhere they go. If we care about the constraints that "ants should make their decision only using local information" then this approach will work as long as the transitions in the MDP are also *local* in the state space. We can even go further: All our analysis (the convergence and the rate of convergence) relies on the "contraction mapping" property. This suggests that any problem that involves the computation of a contraction mappings[6] on a finite space has a natural (superlinear) ant-like solution: ants move around on this finite space and perform local contraction updates.

The ant model we have presented is closely related to previously pub-

[6]Contraction mappings can for instance appear in zero crossing problems, constrained and unconstrained optimization [15].

lished algorithms for the foraging problem. In [20], Simonin presents an algorithm that computes shortest path using an asynchronous implementation of the Bellman-Ford algorithm: the local update is of the form $U(x) \leftarrow 1 + \min_{x' \in \mathcal{N}(x)} U(x')$. Modulo the variable change $U \leftrightarrow \frac{\log(\Phi)}{\log(\beta)}$ this is equivalent to our pheromone update with $\alpha = 1$ expressed in equation 4. In [17], Panait & Luke address the foraging problem and identify, as we proposed, the value function of control problems as pheromone rates. They use an asynchronous version of Value Iteration as well as variants of algorithms from reinforcement learning [23] (which is also formulated through MDPs) to solve the optimal control problem. Furthermore, they present many experiments which illustrate that these models are robust in environments with obstacles and where the positions of the nest and the food source vary over time. As in our model, the authors use multiple pheromones trails for the different paths to find.

There are however significant differences between these works and ours:

1. In the previous published works, the authors only focus on the foraging problem whereas we have just argued that in principle, any optimal control problem (and any computation of contraction mapping) could be tackled by ant-like algorithms.

2. To our knowledge, there are no arguments concerning the convergence for the models of Panait & Luke [17] whereas an extended version of Simonin's model [20] presents a convergence proof, that uses arguments close to the arguments we gave here [21] (recall that the pheromone is here equivalent to the special case of equation 4).

3. Last but not least, the super-linearity of the convergence rate is experimentally observed by Simonin [20], but it is not addressed theoretically.

7 Conclusion

Swarm intelligence algorithms are usually difficult to analyze, and many works of the field rely on extensive experimental simulations. In this paper, we have described several models where many simple interacting agents collectively solve a non trivial task, and for which we could derive an analysis in terms of convergence; to our knowledege, our work is also the first that addresses analytically the rate of convergence with respect to the size of the population. Our analysis, based on the idea that the ant population computes the fixed point of a contraction mapping, explains many results

obtained through simulations, among which the potential superlinearity of the rate of convergence. We have showed how we could extend this model in order to tackle continuous time and space foraging problems and argued that any problem that involves the computation of a contraction mapping fixed point on a finite space has a natural similar (potentially superlinear) ant-like solution. We hope that the methodology we have developped, among which the notion of contraction plays a crucial role, will be useful to the community for analysing and building new Swarm intelligence algorithms. We also believe that such "contracting" ant-like algorithms may be of practical interest in real world decentralized applications.

References

[1] D. Aldous and J. Fill. *Reversible Markov Chains and Random Walks on Graphs*. Monograph in preparation, 1996.

[2] G. Beni and J. Wang. Swarm intelligence. In RSJ press., editor, *Seventh Annual Meeting of the Robotics Society of Japan*, pages 425–428, 1989.

[3] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[4] D.P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice hall, 1989.

[5] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence: from natural to artificial systems*. Oxford University Press, Inc., New York, NY, USA, 1999.

[6] A. Boumaza and B. Scherrer. Optimal control subsumes harmonic control. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 07)*, pages 2841–2846, 2007.

[7] C.I. Connolly and R. Grupen. On the applications of harmonic functions to robotics. *Journal of Robotic and Systems*, 10(7):931–946, October 1993.

[8] J. L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels. The self-organizing exploratory pattern of the Argentine Ant. *Insect Behavior*, 3:159–168, 1990.

[9] J. L. Deneubourg, S. Goss, and N. R. Franks. The blind leading the blind: Modeling chemically mediated army ant raid patterns. *Insect Behavior*, 2:719–725, 1989.

[10] Marco Dorigo and Christian Blum. Ant colony optimization theory: a survey. *Theoretical Computer Science*, 344(2-3):243–278, 2005.

[11] U. Feige and Y. Rabinovich. Deterministic approximation of the cover time. *Random Struct. Algorithms*, 23(1):1–22, 2003.

[12] S. Guerin and D. Kunkle. Emergence of constraint in self-organizing systems. *Nonlinear Dynamics, Psychology, and Life Sciences*, 8(2):131–146, 2004.

[13] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, 1988.

[14] H.J. Kushner and P.G. Dupuis. *Numerical Methods for Stochastic Control Problems in Continuous Time*. Springer Verlag, 1992.

[15] D. Luenberger. *Linear and nonlinear programming*. Addison-Wesley, New York, 1989.

[16] L. Panait and S. Luke. Ant foraging revisited. In *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE-IX)*, 2004.

[17] L. Panait and S. Luke. A pheromone-based utility model for collaborative foraging. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, 2004.

[18] M. Puterman. *Markov Decision Processes*. Wiley, New York, 1994.

[19] M. Resnik. *Turtles termites and traffic jams*. MIT Press, 1994.

[20] O. Simonin. Construction of numerical potential fields with reactive agents. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, 2005.

[21] O. Simonin, F. Charpillet, and E. Thierry. Collective construction of numerical potential fields for the foraging problem. Technical Report RR-6171, INRIA, 2007.

[22] R.S. Sutton. Artificial intelligence as a control problem: Comments on the relationship between machine learning and intelligent control. In *IEEE International Symposium on Intelligent Control*, pages 500–507, 1988.

[23] R.S. Sutton and A.G. Barto. *Reinforcement Learning, An introduction.* BradFord Book. The MIT Press, 1998.

[24] M. Wodrich and G. Bilchev. Cooperative distributed search: The ants' way. *Control and Cybernetics*, 26, 1997.